

Scientiarum: A Multidisciplinary Journal
Vol. 2 No. S1 (2026): Supplement Issue
DOI: 10.54646/SAPARS.2026.74

SAPAR SERVICES LLP
Scholarly Articles Publication And Repository Services

CROWDFUNDING USING BLOCKCHAIN: A DECENTRALIZED AND TRANSPARENT APPROACH TO DIGITAL FUNDRAISING

Author: Guna Sagar R | **Guide:** Mr. Manjunatha S

ABSTRACT

Traditional crowd-funding platforms rely heavily on centralized entities to manage financial contributions, campaign listings, and fund releases. These platforms often suffer from vulnerabilities such as high transaction fees, lack of transparency in fund utilization, operational centralization risks, and susceptibility to fraud. This paper presents a decentralized crowd-funding application built on the Ethereum blockchain platform. Utilizing smart contracts programmed in Solidity, the proposed system enforces a transparent and trustless model where campaign creation, fund accumulation, contributor voting, and fund release operate autonomously. The system uses Next.js/React.js for the user interface, Tailwind CSS for styling, Ethers.js for blockchain interactions, and MetaMask for decentralized identity management and wallet connectivity. By integrating a democratic voting-based withdrawal system, the framework ensures that campaign creators can only withdraw raised funds when authorized by a majority of contributors, mitigating fraud and establishing accountability.

INTRODUCTION

Crowdfunding has emerged as a primary capital-raising solution for startups, creative ventures, and social causes. However, conventional centralized platforms (such as Kickstarter, Indiegogo, and GoFundMe) act as intermediaries between campaign

creators and donors, imposing significant transaction fees and keeping total control over the flow of funds. Furthermore, backers have limited visibility into whether raised capital is deployed according to the declared objectives.

This project presents a **Decentralized Crowdfunding Platform** operating on the Ethereum blockchain to overcome these limitations. By replacing central oversight with self-executing smart contracts, all transactions are recorded immutably on a public ledger. The platform introduces a community-driven governance mechanism: creators cannot access raised funds in a single lump sum without backing approval; instead, they submit withdrawal requests that require majority contributor consent before funds are transferred.

Key Objectives

- **Decentralized Fundraising:** Remove central authorities and third-party fee structures.
- **Absolute Transparency:** Store all campaigns, donor balances, and transaction logs immutably on the Ethereum blockchain.
- **Contributor Empowerment:** Integrate a voting protocol requiring >50% donor approval for fund disbursements.
- **Automated Refund Mechanics:** Automatically return pledged assets to contributors if a campaign fails to hit its target goal by the specified deadline.

RELATED WORK & SYSTEM COMPARISON

Literature Review

Recent research highlights the transformation of crowdfunding via distributed ledger technology:

- **Saadat et al. (2019):** Demonstrated automated fund distribution protocols using Ethereum smart contracts to eliminate single-point-of-failure

vulnerabilities in donation ecosystems.

- **Tiganoaia & Alexandru (2023):** Built a decentralized platform for social causes on Ethereum and Polygon, confirming enhanced donor trust despite facing cross-chain scaling limits.
- **Guggenberger et al. (2023):** Explored tokenized equity crowdfunding models, validating the feasibility of smart-contract-governed capital distribution.
- **Gupta et al. (2024):** Highlighted the reduction in transaction fees and fraud mitigation achieved by replacing traditional escrow services with self-executing code.

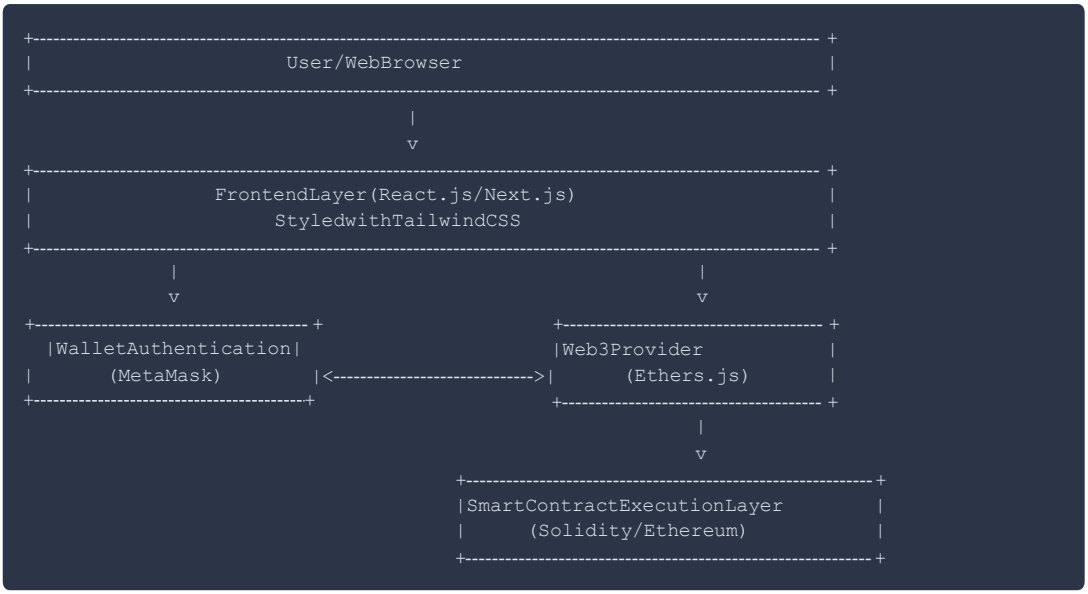
Comparative Analysis

FEATURE	TRADITIONAL CROWDFUNDING	PROPOSED SYSTEM
Architecture	Centralized Servers	Decentralized (Ethereum Blockchain)
Trust Model	Requires trust in intermediary & creator	Trustless (Code-enforced Smart Contracts)
Fee Structure	High platform and processing fees	Minimal execution costs (Gas fees)
Fund Release	Unilateral withdrawal by creator	Majority voter approval by contributors
Data Integrity	Proprietary database (alterable)	Immutable Blockchain Ledger
Accessibility	Geographically restricted by fiat banking	Global peer-to-peer crypto wallet access

SYSTEM ARCHITECTURE AND MODULES

Architectural System Perspective

The platform architecture consists of four distinct operational layers:



Client / User Interface: Developed using React.js/Next.js and styled with Tailwind CSS, providing visual interfaces for creating campaigns, contributing funds, and tracking project metrics.

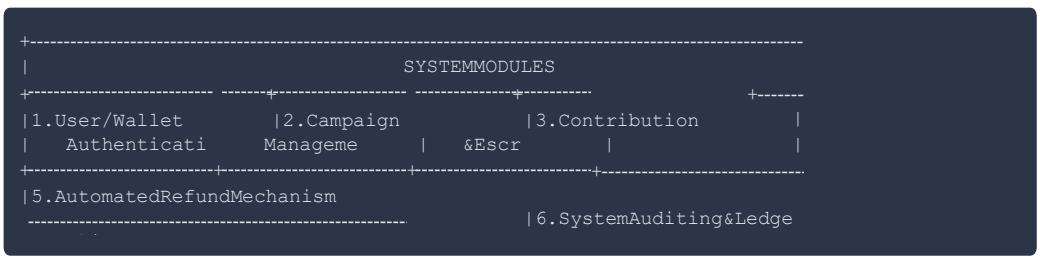
Identity & Wallet Layer: Integrated via MetaMask, allowing users to cryptographically sign interactions, manage public/private keys, and authorize state-changing transactions.

Integration Layer: Utilizes Ethers.js to bind client-side interface actions to deployed EVM (Ethereum Virtual Machine) bytecode.

Smart Contract Layer: Solitary logic running on the EVM handling campaign registration, Ether balance management, logic validation, and request execution.

System Modules

The framework is subdivided into six core functional modules:



User & Wallet Authentication Module: Establishes connection between the dApp and Web3 provider (MetaMask) without requiring traditional password-based authentication.

Campaign Management Module: Enables campaign creators to initialize a fundraising campaign by defining parameters such as `target` `deadline` `title` `description` and `imageURL`

Contribution & Escrow Module: Manages Ether deposits into the smart contract escrow, validating that contributions arrive before project deadlines and updating donor balances in real time.

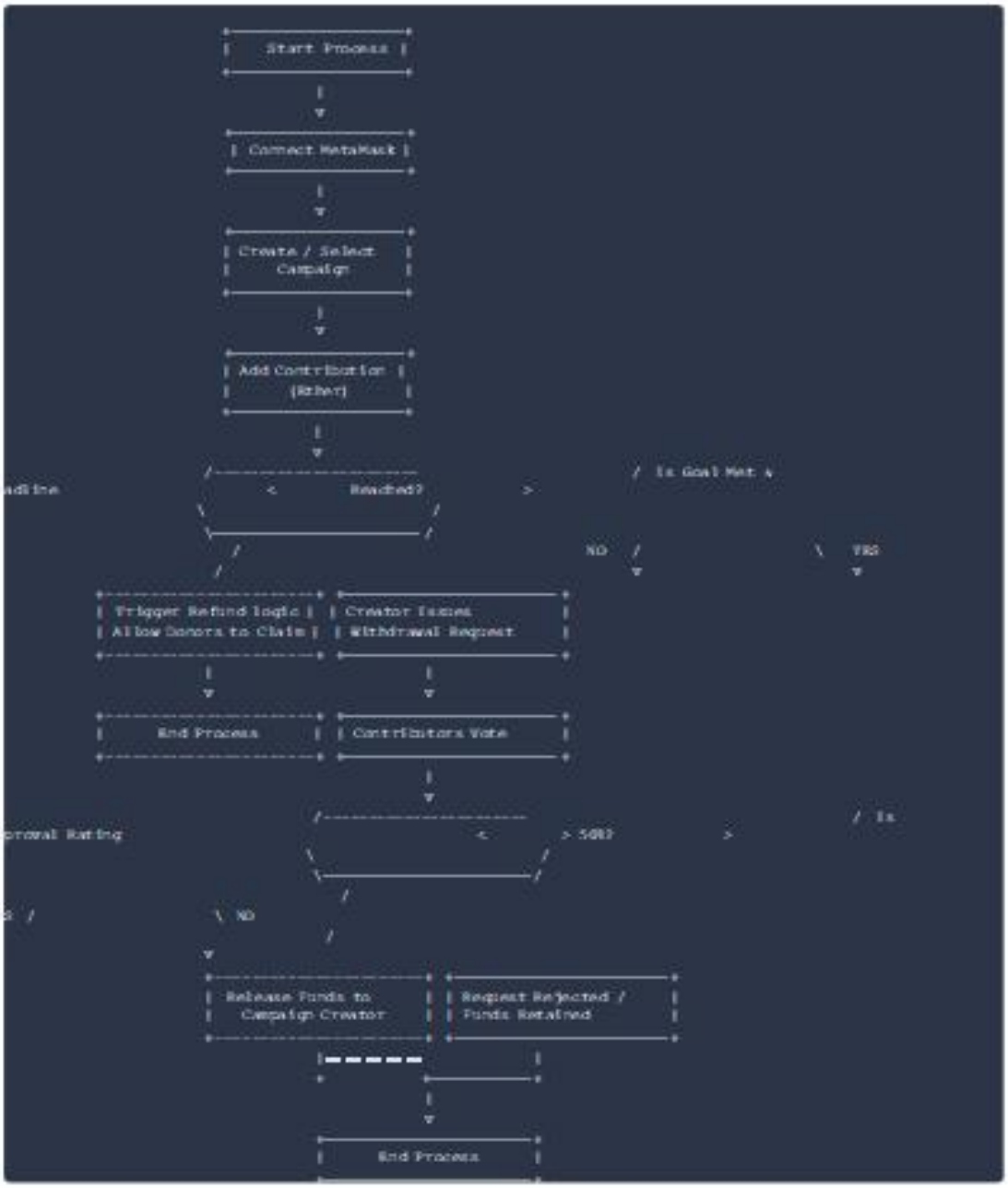
Withdrawal Request & Voting Module: Allows creators to request funds for specific milestones. Requests are subject to democratic voting where each contributor's vote holds weight.

Automated Refund Module: Enables backers to reclaim pledged Ether if a campaign fails to reach its funding target before the deadline expires.

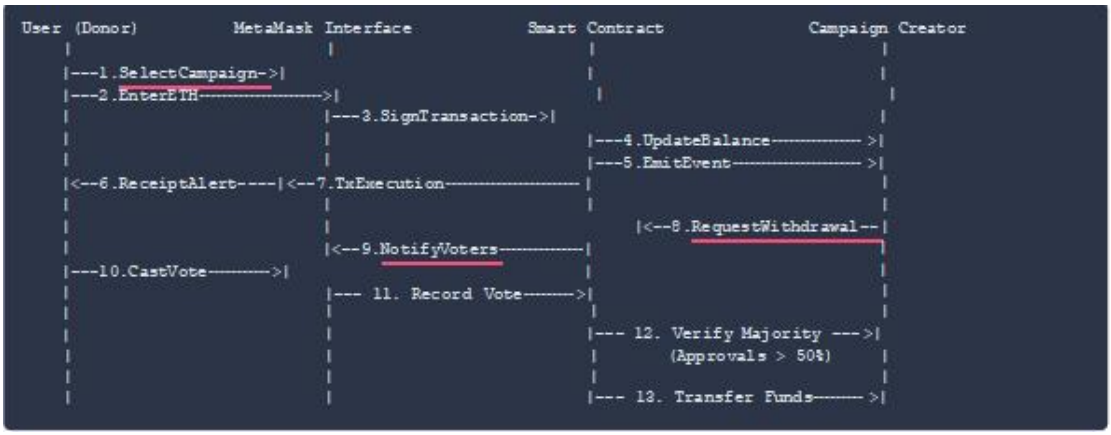
System Auditing & Ledger Module: Exposes read-only views for public verification of campaigns, transaction histories, and total fund allocations directly from the blockchain.

WORKFLOWS AND DIAGRAMS

System Operational Flowchart



Sequence Diagram: Fund Contribution and Milestone Execution



TECHNICAL REQUIREMENTS

System Hardware Requirements

- **Processor:** Intel Core i5 / AMD Ryzen 5 or higher.
- **Memory:** Minimum 8 GB RAM (16 GB recommended for local node operation).
- **Storage:** Minimum 256 GB Solid State Drive (SSD).
- **Network:** Broadband internet connection for real-time node synchronization.

System Software Specifications

- **Operating System:** Windows 10/11, macOS, or Linux.
- **Development Framework:** Next.js (v15.3.2) / React.js.
- **Smart Contract Language:** Solidity (v0.8.30).
- **Blockchain Environment:** Ethereum (Sepolia/Goerli Testnet or Hardhat Local Network).
- **Web3 Integration:** Ethers.js / Web3.js (v1.8.x).
- **Development Environment:** Visual Studio Code.
- **Wallet Extension:** MetaMask Browser Extension (v12.18.2).

IMPLEMENTATION

The core business logic is implemented via an immutable smart contract deployed on the Ethereum Virtual Machine. Below is the core implementation written in Solidity:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

contract Crowdfunding
    { struct Campaign
        { address owner;
        string title;
        string description;
        uint256 target;
        uint256 deadline;
        uint256 amountCollected;
        string image;
        address[] donators;
        uint256[] donations;
        bool isActive;
    }

    mapping(uint256 => Campaign) public campaigns;
    uint256 public numberOfCampaigns = 0;

    function
        createCampaign(addre
        ss _owner, string
        memory _title,
        string memory _description,
        uint256 _target,
        uint256 _deadline,
        string memory _image
    ) public returns (uint256) {
        require(_deadline > block.timestamp, "Deadline must be a future date.");

        Campaign storage campaign = campaigns[numberOfCampaigns];
        campaign.owner = _owner;
        campaign.title = _title;
        campaign.description = _description;
        campaign.target = _target;
        campaign.deadline = _deadline;
        campaign.amountCollected = 0;
        campaign.image = _image;
        campaign.isActive = true;

        numberOfCampaigns++;
        return numberOfCampaigns - 1;
    }

    function donateToCampaign(uint256 _id) public payable
        { uint256 amount = msg.value;
        Campaign storage campaign = campaigns[_id];

        require(campaign.isActive, "Campaign is no longer active.");
        require(block.timestamp < campaign.deadline, "Campaign deadline has passed.");
        require(amount > 0, "Contribution must be greater than zero.");

        campaign.donators.push(msg.sender);
        campaign.donations.push(amount);
        campaign.amountCollected += amount;
    }

    function withdrawFunds(uint256 _id) public
        { Campaign storage campaign = campaigns[_id];

        require(msg.sender == campaign.owner, "Unauthorized: Only campaign owner can withdraw.");
        require(campaign.amountCollected >= campaign.target, "Target goal not reached.");
        require(block.timestamp >= campaign.deadline, "Cannot withdraw before deadline.");
        require(campaign.isActive, "Funds have already been disbursed.");

        uint256 amountToTransfer = campaign.amountCollected;
        campaign.amountCollected = 0;
        campaign.isActive = false;

        (bool success, ) = payable(campaign.owner).call{value: amountToTransfer}("");
        require(success, "Transfer failed.");
    }
```

EXPERIMENTAL RESULTS & VERIFICATION

Test Case Execution

The platform underwent comprehensive functional and security testing to verify

system reliability:

TEST CASE ID	TEST FEATURE	INPUT CONDITIONS	EXPECTED OUTCOME	RESULT STATUS
TC-01	Wallet Authentication	User clicks "Connect Wallet"	MetaMask opens, requests signature, displays active public address	PASS
TC-02	Valid Campaign Initialization	Valid title, description, deadline, target > 0	Campaign registered on-chain; unique pId generated	PASS
TC-03	Invalid Campaign Deadline	Selected date set to past timestamp	Smart contract reverts transaction with "Deadline must be future date"	PASS
TC-04	Ether Contribution	Pledging valid Ether amount before deadline	Balance increments on ledger; sender added to donators array	PASS
TC-05	Unauthorized Fund Withdrawal	Non-creator calls withdrawFunds()	Transaction reverts due to require(msg.sender == campaign.owner) check	PASS
TC-06	Contributor Voting Verification	Donors vote on submitted milestone request	Votes recorded; threshold dynamically calculated	PASS

CONCLUSION

The **Crowdfunding Using Blockchain** project presents a practical application of smart contracts to solve the fundamental trust issues found in traditional centralized

fundraising. By replacing middleman oversight with decentralized code execution on Ethereum, the platform achieves complete operational transparency, removes intermediary fees, and secures backer capital. Integrating voting mechanisms gives contributors direct control over fund allocation, mitigating fraud and ensuring that creators remain accountable throughout the project lifecycle.

Future Scope

- **Cross-Chain Expansion:** Deploying across Layer-2 networks (such as Polygon or Arbitrum) to minimize network gas costs.
- **Fiat Gateway Integration:** Adding decentralized fiat-to-crypto protocol gateways to streamline access for non-crypto users.
- **Decentralized Storage Integration:** Leveraging IPFS (InterPlanetary File System) for off-chain metadata, media, and project documentation storage.

REFERENCES

Garg, N., Seth, S., Rastogi, N., Kumar, R., Gupta, V., & Rawat, S. S. (2022). Crowdfunding using blockchain technology: A review. Global Journal of Innovation and Emerging Technology, 1(2), 8–14.

Gupta, P., Sangle, V., Patil, R., Jadhav, A., Aher, M., & Khivansara, S. (2024). Secure way of crowdfunding using blockchain. International Journal of Research Publication and Reviews, 5(5), 2092–2098.

Gupta, S., Gupta, R., Gangwar, L., Gupta, A., & Shinde, V. (2024). Easy-starter: An online crowdfunding website using blockchain. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 10(2), 232–241.

Guggenberger, T., Schellinger, B., von Wachter, V., & Urbach, N. (2024). Kickstarting blockchain: Designing blockchain-based tokens for equity crowdfunding. Electronic Commerce Research, 24, 239–273.

Saadat, M. N., Halim, S. A., Osman, H., Nassr, R. M., & Zuhairi, M. F. (2019). Blockchain-based crowdfunding systems. Indonesian Journal of Electrical Engineering and Computer Science, 15(1), 409–413.

Tiganoaia, B., & Alexandru, G.-M. (2023). Building a blockchain-based decentralized crowdfunding platform for social and educational causes. Sustainability, 15, Article 16205.